Interactive Programs

Graphical User Interfaces

CS112 Scientific Computation
Department of Computer Science
Wellesley College

Properties of graphics objects

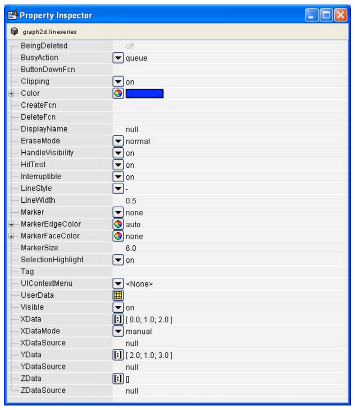
All plotting and graphics functions create graphic objects

Each graphics object is identified by a unique number call a handle that can be assigned to a variable:

```
p1 = plot([0  1  2],  [2  1  3]);
```

Graphics objects have properties that control their appearance on the screen and can be viewed with the Property Inspector:

```
inspect(p1)
```



graph2d.lineseries

BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
Clipping	on
Color	🟦
CreateFcn	
DeleteFcn	
DisplayName	null
EraseMode	normal
HandleVisibility	on
HitTest	on
Interruptible	on
LineStyle	-
LineWidth	0.5
Marker	none
MarkerEdgeColor	auto
MarkerFaceColor	none
MarkerSize	6.0
SelectionHighlight	on
Tag	
UIContextMenu	<None>
UserData	
Visible	on
XData	[0.0; 1.0; 2.0]
XDataMode	manual
XDataSource	null
YData	[2.0; 1.0; 3.0]
YDataSource	null
ZData	[]
ZDataSource	null

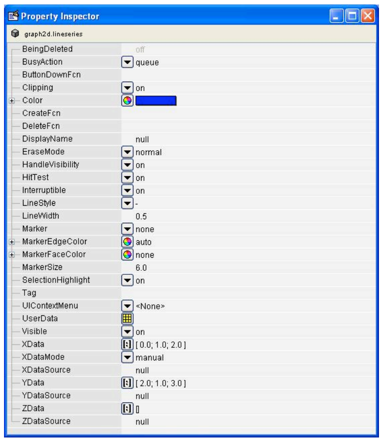
Accessing properties with MATLAB code

Graphics object properties can be accessed with the get function:

get(object, property)

For example,

```
>> get(p1, 'LineWidth');
0.5
```



Property Inspector

graph2d.lineseries

BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
Clipping	on
Color	🟦
CreateFcn	
DeleteFcn	
DisplayName	null
EraseMode	normal
HandleVisibility	on
HitTest	on
Interruptible	on
LineStyle	-
LineWidth	0.5
Marker	none
MarkerEdgeColor	auto
MarkerFaceColor	none
MarkerSize	6.0
SelectionHighlight	on
Tag	
UIContextMenu	<None>
UserData	
Visible	on
XData	[0.0; 1.0; 2.0]
XDataMode	manual
XDataSource	null
YData	[2.0; 1.0; 3.0]
YDataSource	null
ZData	[]
ZDataSource	null

Graphics object properties can be set* by

... editing the value in the Property Inspector window

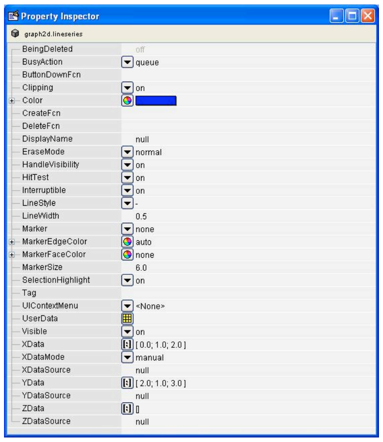
... specifying the property name and value when creating the graphics object:

```
p1 = plot([0 1 2], [2 1 3],  'LineWidth', 1);
```

... using the set function:

set(object, property, value)

```
>> set(p1, 'LineWidth', 1);
```



Property Inspector

graph2d.lineseries

BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
Clipping	on
Color	🟦
CreateFcn	
DeleteFcn	
DisplayName	null
EraseMode	normal
HandleVisibility	on
HitTest	on
Interruptible	on
LineStyle	-
LineWidth	0.5
Marker	none
MarkerEdgeColor	auto
MarkerFaceColor	none
MarkerSize	6.0
SelectionHighlight	on
Tag	
UIContextMenu	<None>
UserData	
Visible	on
XData	[0.0; 1.0; 2.0]
XDataMode	manual
XDataSource	null
YData	[2.0; 1.0; 3.0]
YDataSource	null
ZData	[]
ZDataSource	null

* true for any graphics function, e.g. figure, fill, scatter, surf

Subfunctions

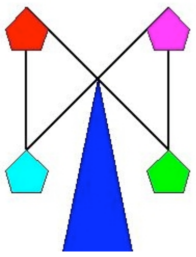
An M-file can only contain one function that can be called from the Command Window or from another code file

This function must be placed at the beginning of the file and its name must be the same as the file name

Other *subfunctions* can be defined in an M-File, but can only be called by functions in the same M-File

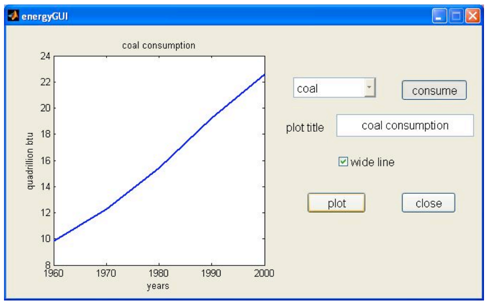
Subfunctions for a ferris wheel movie

```
function ferrisWheel
% displays an animation of a rotating ferris wheel
for frame = 1:36
    drawBase;
    hold on
    spokeCoords = drawWheel(10*frame);
    drawCars(spokeCoords);
    hold off
end

function drawBase
% draw the blue base of the ferris wheel

function spokeCoords = drawWheel (angle)
% draw the black spokes at the input angle and return
% the coordinates of the endpoints of the spokes

function drawCars (spokeCoords)
% draw a colored car at each location in spokeCoords
```



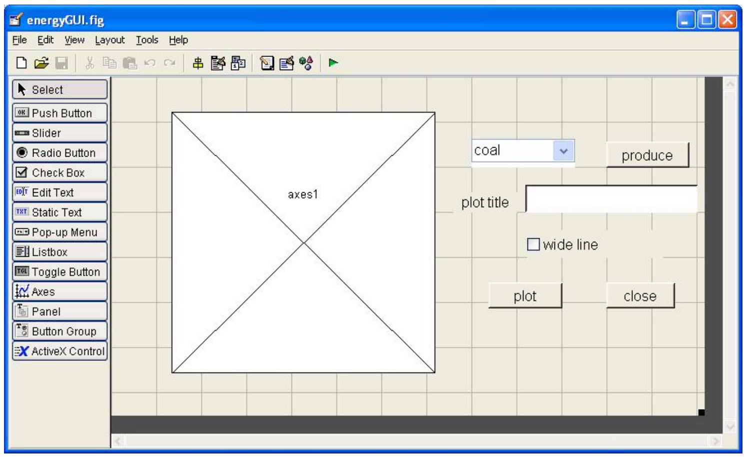
Graphical User Interface (GUI)

For our programs to date, we called a function or script from the Command Window and the program was executed with minimal input from the user



GUI-based programs put the user in the driver's seat through interactions with components of a graphical display

MATLAB's Graphical User Interface Development Environment (GUIDE)

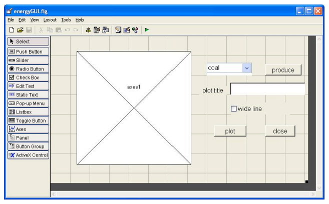


Saving the GUI layout

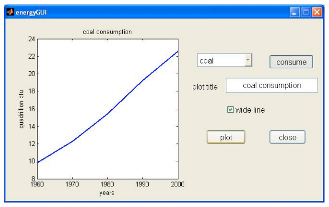
When our energyGUI layout is saved the first time, MATLAB generates two files:

energyGUI.fig: Layout Editor window with the developing GUI, which can be modified later by entering

```
>> guide energyGUI.fig
```



energyGUI.m: file that contains code to create the GUI display



Functions defined in energyGUI.m

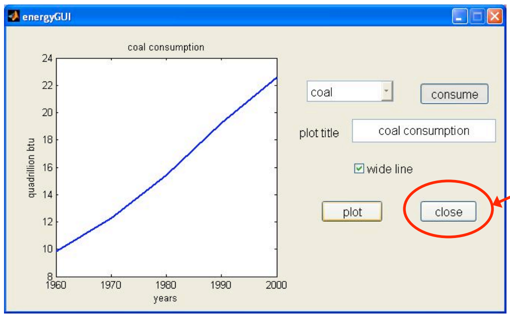
energyGUI: top-level function at the beginning of the file that is called from the Command Window. This function initializes the GUI program and opens the GUI window. *We will not modify this function*

energyGUI_OpeningFcn: executed just before the GUI window is made visible. We will modify this function to set up data for the program

energyGUI_OutputFcn: returns outputs to the Command Window. *We will not modify this function*

Callback functions

For each component, the header of a Callback function is created. These functions are invoked automatically when the user interacts with the corresponding component



Invokes closeButton_Callback when clicked by user

Inputs to GUI functions

hObject is a number, the *graphics handle*, that uniquely identifies the GUI component and its associated properties

eventdata is not used in the current version of MATLAB

handles is a structure that contains information that needs to be shared between functions in this file. Initially it contains a field for each GUI component created, using Tag property as name:

$$\left.\begin{array}{l}\texttt{handles.axes1}\\\texttt{handles.sourceMenu}\\\texttt{handles.sourceToggle}\\\texttt{handles.titleLabel}\\\texttt{handles.titleBox}\\\texttt{handles.widthCheckbox}\\\texttt{handles.plotButton}\\\texttt{handles.closeButton}\\\texttt{handles.figure1}\end{array}\right\}\text{Value of each field is the graphics handle for that component}$$

Adding actions

```
function energyGUI_OpeningFcn (hObject, eventdata, handles, varargin)
% setup data to use for plotting
[handles.years  handles.produce  handles.consume] = setupEnergy;

function sourceToggle_Callback (hObject, eventdata, handles)
% use state of toggle button to set text label on button
if (get(hObject, 'Value') == 0)
    set(hObject, 'String', 'produce');
else
    set(hObject, 'String' , 'consume');
end
guidata(hObject, handles);
```

copy changes to global handles structure

More action

```
function plotButton_Callback (hObject, eventdata, handles)
% setup data source requested by user from state of toggle button
if (get(handles.sourceToggle, 'Value') == 0)
    dataSource = handles.produce;
else
    dataSource = handles.consume;
end
% get index of selected energy source
sourceIndex = get(handles.sourceMenu, 'Value');
% use state of checkbox to determine line width
linewidth = get(handles.widthCheckbox, 'Value') + 1;
% plot the data with the requested properties
plot(handles.years, dataSource(sourceIndex, :), 'Linewidth', linewidth);
xlabel('years')
ylabel('quadrillion btu')
title(get(handles.titleBox, 'String'))
```

Time for you to leave

```
function closeButton_Callback (hObject, eventdata, handles)
% close GUI figure window
delete(handles.figure1);
```